A Graph-incorporated Latent Factor Analysis Model for High-dimensional and Sparse Data

Di Wu, *Member, IEEE*, Yi He, *Member, IEEE*, and Xin Luo, *Senior Member, IEEE*

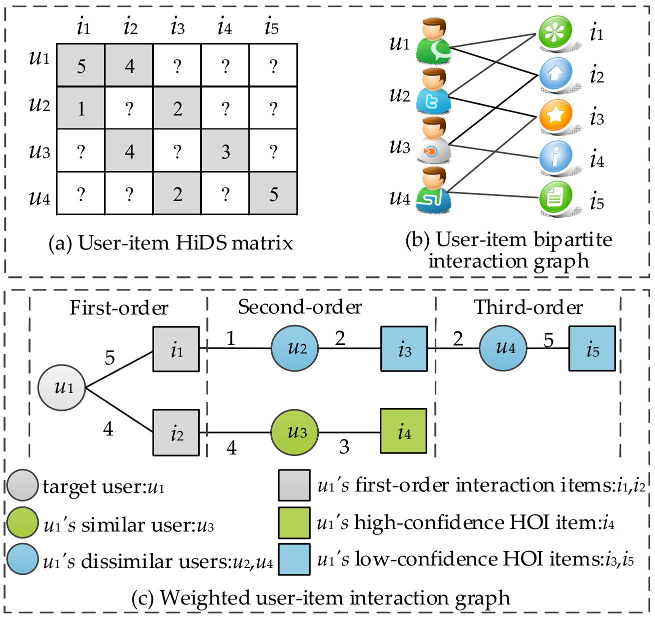
Abstract—A High-dimensional and sparse (HiDS) matrix is frequently encountered in big data-related applications such as e-commerce systems or wireless sensor networks. It is of great significance to perform highly accurate representation learning on an HiDS matrix due to the great desires of extracting latent knowledge from it. Latent factor analysis (LFA), which represents an HiDS matrix by learning the low-rank embeddings based on its observed entries only, is one of the most effective and efficient approaches to this issue. However, most existing LFA-based models directly perform such embeddings on an HiDS matrix without exploiting its hidden graph structures, resulting in accuracy loss. To aid this issue, this paper proposes a graph-incorporated latent factor analysis (GLFA) model. It adopts two-fold ideas: 1) a graph is constructed for identifying the hidden high-order interaction (HOI) among nodes described by an HiDS matrix, and 2) a recurrent LFA structure is carefully designed with the incorporation of HOI, thereby improving the representation learning ability of a resultant model. Experimental results on three real-world datasets demonstrate that GLFA outperforms six state-of-the-art models in predicting the missing data of an HiDS matrix, which evidently supports its strong representation learning ability to HiDS data.

Index Terms—High-dimensional and Sparse Matrix, Latent Factor Analysis, Missing Data Estimation, Graph Structure, Graph Representation Learning.

1 Introduction

In industrial applications, matrices are widely adopted to describe the relationships between two types of entities. In many big-data-related applications like bioinformatics networks [1], e-commerce systems [2], wireless sensor networks [3], and intelligent transportation [4], the relationships among numerous entities are unlikely to be fully observed in practice [5]. As a result, matrices from these applications are usually high-dimensional and sparse (HiDS). Although they are highly sparse, they contain rich latent knowledge and patterns [6, 7], e.g., users' potential preferences on items in e-commerce systems. Therefore, precisely extracting valuable knowledge and patterns from such HiDS matrices becomes a hot yet thorny issue in industrial applications [8].

Recently, latent factor analysis (LFA) [6, 7] has become one of the most popular and successful approaches to this issue. Given an HiDS matrix, an LFA model aims to learn two low-rank embedding matrices to estimate missing entries based on observed ones only [9]. However, most existing LFA-based models learn the embeddings from an HiDS matrix directly without exploiting its hidden graph structures [10]; thereby, the learned embeddings may not be sufficient to represent it accurately. Fig. 1 gives an example to illustrate the hidden graph structures of an HiDS matrix.



(a) User-item HiDS matrix
(b) User-item bipartite interaction graph
First-order
Second-order
Third-order
target user: u_1
u_1's similar user: u_3
u_1's dissimilar users: u_2, u_4
u_1's first-order interaction items: i_1, i_2
u_1's high-confidence HOI item: i_4
u_1's low-confidence HOI items: i_3, i_5
(c) Weighted user-item interaction graph

Fig. 1. A motivation example: an HiDS matrix hides a graph with high-order interaction (HOI).

Example. Fig. 1(a) is a typical HiDS matrix from an e-commerce system with four users (u_1, u_2, u_3, u_4) and five items (i_1, i_2, i_3, i_4, i_5), where grey entries denote the observed users' preferences on corresponding items. From Fig. 1(a), the user-item bipartite interaction graph can be drawn as shown in Fig. 1(b), where solid lines denote the interactions between

This work is supported by the National Natural Science Foundation of China under grants 62176070 and 62272078.

- *D. Wu and X. Luo are with the College of Computer and Information Science, Southwest University, Chongqing 400715, China, (e-mail: wudi.cigit@gmail.com, luoxin21@gmail.com).*
- *Y. He is with the Old Dominion University, Norfolk, Virginia 23529, USA. (yihe@cs.odu.edu).*
- *Corresponding author: X. Luo.*

corresponding users and items. In particular, if considering the connection weights (the values of preferences) for Fig. 1(b), the weighted user-item interaction graph can be constructed as shown in Fig. 1(c). Fig. 1(c) clearly shows that there are some high-order interactions (HOIs) among nodes. For example, for a target user u_1, i_3 and i_4 are its second-order interaction items, and i_5 is its third-order interaction item. Further analysis, i_4 is u_1's high-confidence HOI item because u_3 (connected with i_4) rates the value four on i_2, the same as that rated by u_1. Besides, i_3 and i_5 are u_1's low-confidence HOI items because their connected u_2 rates a different value with u_1 on i_1. Note that the terms of *HOI* and *high/low-confidence HOI* are formally defined in Section 4.1.

This example shows that an HiDS matrix hides the graph structures that contain many HOIs among nodes. But, the problem is how to effectively exploit these HOIs information to enhance an LFA model's representation learning ability to an HiDS matrix.

Recently, Zhou *et al*. propose the *Deep Forest* to alleviate some drawbacks of deep neural networks (DNNs) [11]. Its principle is to gradually enhance a basic model by enriching its inputs via a carefully designed multilayer learning structure. Each layer is a basic model, and its additional inputs come from the prediction outputs of its preceding basic model. From this point of view, it becomes possible to gradually enhance an LFA model by recurrently enriching its inputs via a carefully designed recurrent (or multilayered) learning structure, where the additional inputs come from the prediction outputs of a well-trained LFA model. In particular, if the aforementioned HOI information can be recurrently incorporated into such prediction outputs of an LFA model, an LFA model's representation learning ability to an HiDS matrix would be greatly enhanced.

To this end, this paper proposes a graph-incorporated latent factor (GLFA) model. It consists of two main parts. One part is to construct a graph from a given HiDS matrix for identifying its hidden HOI among nodes. The other is to design a recurrent LFA structure to gradually enhance an LFA model by incorporating the identified HOI information into the embedding learning process. Experimental results on three real-world datasets verify that the proposed GLFA model outperforms six state-of-the-art models in representing HiDS data, including four LFA-based and two DNNs-based models.

The main contributions of this work include:

- A GLFA model that can accurately represent an HiDS matrix to predict its missing data is proposed;
- Algorithm design and analysis are given for the proposed GLFA model;
- Extensive experiments on three real-world datasets are conducted to evaluate the proposed GLFA model, including comparing it with eight state-of-the-art models and analyzing its characteristics.

The rest of this paper is organized as follows. Section 2 analyzes the related work. Section 3 states the preliminaries. Section 4 presents the proposed GLFA model. Section 5 provides the empirical studies. Finally, Section 6 concludes this article.

2 Related Work

Since LFA has achieved great success in the high-profile competition of the Netflix Prize [6], LFA-based models were springing up in the past decade. Various sophisticated LFA-based models have been proposed for analyzing HiDS data, including dual-regularization-based [12], non-negative constraint [5], generalized momentum-based [13], probabilistic [14], neighborhood-and-location integrated [15], joint recommendation [16], confidence-driven [17], content features-based [18], deep-structure-based [19], generalized non-negative based [20], and data-characteristic-aware [21] ones. However, given an HiDS matrix, the existing LFA-based models ignore its hidden graph structures that contain rich latent knowledge and patterns [10] during their embedding learning process. Sharply different from these models, the proposed GLFA model can incorporate the HOI information of a graph constructed from an HiDS matrix into its embedding learning process via a recurrent LFA structure, thereby achieving the accurate representation of an HiDS matrix.

Recently, DNN has attracted much attention in analyzing HiDS data due to their powerful representation learning ability [22], especially in recommender systems. Zhang *et al*. provided a detailed review *w.r.t.* DNNs-based recommender systems [23]. To date, many sophisticated DNNs-based models have been proposed. Representative models include autoencoder-based [24], neural collaborative filtering-based [22], variational autoencoder-based [25], hybrid autoencoder-based [26], neural factorization-based [27], federated meta-learning-based [28], latent relevant metric learning-based [29], and deep-rating and review-neural-network-based [30] ones.

Moreover, regarding exploiting the hidden graph structures of an HiDS matrix, some graph neural networks (GNNs)-based models have been also proposed. Berg et al. proposed a graph autoencoder-based model [10] based on differentiable messages passing on the bipartite interaction graph. Later, Zhang *et al*. proposed an inductive graph-based matrix completion model based purely on 1-hop subgraphs around pairs generated from the HiDS matrix [31]. Wang *et al*. proposed a multi-component graph convolutional collaborative filtering approach to distinguish the latent purchasing motivations underneath the HiDS matrix[32]. Zhang *et al*. proposed a graph attention multi-layer perceptron to capture the underlying correlations between different scales of graph knowledge [42]. Besides, some researchers proposed to employ GNNs to represent the HiDS matrix for item ranking problems of recommender systems, such as neural graph collaborative filtering [33], light neural graph collaborative filtering [34], self-supervised graph learning [35], GNNs-based geographical attentive recommendation [43], and graph-based regularization approach [44]. Wu *et al*. provided a detailed review *w.r.t.* GNNs-based recommender systems [45].

Notably, although the DNNs- and GNNs-based models show some promising advances in processing HiDS data, the proposed GLFA model is significantly different from them as follows:

- Their computational complexity is high because they usually take complete data rather than known data of an HiDS matrix as input [11]. In comparison, GLFA has higher computational efficiency as it is trained only on observed entries of an HiDS matrix.
- To discover the hidden graph structure, some require both the HiDS matrix and the additional features of entities, like text comments [15, 33]. While GLFA fully exploits the hidden graph structures only based on the HiDS matrix.
- S. Rendle et al. demonstrate that the LFA model is still highly competitive with them [36]. By incorporating the hidden graph structures into the LFA model, GLFA achieves better performance than them (please refer to Section 5.2 Performance Comparison).

3 Preliminaries

3.1 Symbols and Nomenclatures

The adopted main symbols and nomenclatures of this study are summarized in Table 1.

TABLE 1. SYMBOLS AND NOMENCLATURES.

Notation	Explanation
HiDS	High-dimensional and sparse (HiDS).
LFA	Latent factor analysis (LFA).
HOI	High-order interaction (HOI).
DNN	Deep neural network (DNN).
GNN	Graph neural network (GNN).
GLFA	graph-incorporated latent factor analysis (GLFA).
U, I	Two types of entity sets, respectively.
u,i	An entity of U and I, respectively, $u \in U$, $i \in I$.
R	An HiDS matrix with $\lvert U \rvert$ rows and $\lvert I \rvert$ columns to record the relationships between U and I.
$r_{u,i}$	An entry of R at u-th row and i-th column.
Ψ, Γ	The sets of observed and unobserved entries of R, respectively.
f	Latent factor dimension.
$\hat{\mathrm{R}}$	R's rank-f approximation with $\lvert U \rvert$ rows and $\lvert I \rvert$ columns.
$\hat{r}_{u,i}$	An entry of $\hat{\mathrm{R}}$ at u-th row and i-th column corresponding to $r_{u,i}$.
n	The n-th recurrent training of GLFA, $n \in \{1, 2, \ldots, N\}$.
$\hat{\mathrm{R}}^n$	R's rank-f approximation at the n-th recurrent training.
$\hat{r}^n_{u,i}$	An entry of $\hat{\mathrm{R}}^n$ at u-th row and i-th column.
X, Y	Two embedding matrices of R with $\lvert U \rvert \times f$ and $\lvert I \rvert \times f$, respectively.
X^n,Y^n	Two embedding matrices of R with $\lvert U \rvert \times f$ and $\lvert I \rvert \times f$ at the n-th recurrent training, respectively.
$x^n_{u,\tau,}$ $y^n_{i,\tau,}$	An entry of X^n and Y^n at u-th row and τ-th column, and i-th row and τ-th column, respectively.
Ω	A $\lvert U \rvert \times \lvert I \rvert$ binary index matrix of R.
$\Omega_{u,i}$	An entry of Ω at u-th row and i-th column.
G	Weighted interaction graph *of* R.
E	Edge set of G.
$e_{u,i}$	Each edge $e_{u,i}$ of E.
p	The p-th high-order interaction of G.
S	High-confidence HOI set of G.
S^n	A subset of S ($S^n \subseteq S$) whose elements are randomly selected from S at the n-th recurrent training.
$\hat{S}^n$	The corresponding prediction set of S^n obtained by $\hat{\mathrm{R}}^n = \mathrm{X}^n \mathrm{Y}^{n\mathrm{T}}$.
Λ	A prediction set of S by previous n-1 times of recurrent training.
Λ^n	A set to store the prediction outputs of the activation function at the n-th recurrent training.
$\bar{r}_{u,i}$	An element of Λ corresponds to unknown $r_{u,i}$.
α	The aggregation coefficient of high-confidence HOIs.
λ	Regularization hyperparameter.
η	The learning rate of stochastic gradient descent.

3.2 HiDS Matrix and LFA Model

***Definition* 1** (*HiDS Matrix*). Given two types of entity sets U and I, R is a $\lvert U \rvert \times \lvert I \rvert$ matrix in which each entry $r_{u,i}$ denotes the relationship between entity u ($u \in U$) and entity i ($i \in I$). Ψ and Γ respectively denote the sets of observed and unobserved entries of R. R is a high-dimensional and sparse (HiDS) matrix with $\lvert \Psi \rvert \ll \lvert \Gamma \rvert$.

***Definition* 2** (*LFA Model*). Given an R, a latent factor analysis (LFA) model is to learn two embedding matrices $\mathrm{X}^{\lvert U \rvert \times f}$ and $\mathrm{Y}^{\lvert I \rvert \times f}$ to make R's rank-f approximation $\hat{\mathrm{R}}$ by minimizing the distances between R and $\hat{\mathrm{R}}$ on Ψ, where f is the embedding dimension with $f \ll \min\{\lvert U \rvert, \lvert I \rvert\}$ and $\hat{\mathrm{R}}$ can be obtained by $\hat{\mathrm{R}} = \mathrm{XY}^{\mathrm{T}}$. $\hat{\mathrm{R}}$'s each entry $\hat{r}_{u,i}$ is the corresponding prediction for $r_{u,i}$.

Definition 2 shows that an LFA model requires to design an objective function to minimize the distances between R and $\hat{\mathrm{R}}$ on Ψ. Commonly, Euclidean is adopted as the metric distance to design such an objective function as follows:

$$\underset{\mathrm{X},\,\mathrm{Y}}{\arg\min}\,\varepsilon(\mathrm{X},\,\mathrm{Y}) = \frac{1}{2}\left\|\Omega \odot \left(\mathrm{R} - \hat{\mathrm{R}}\right)\right\|_{\mathrm{F}}^{2} = \frac{1}{2}\left\|\Omega \odot \left(\mathrm{R} - \mathrm{XY}^{\mathrm{T}}\right)\right\|_{\mathrm{F}}^{2}, \quad (1)$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm of a matrix, $\odot$ denotes the Hadamard product (the component-wise multiplication), and Ω is a $\lvert U \rvert \times \lvert I \rvert$ binary index matrix given by:

$$\Omega_{u,i} = \begin{cases} 1 & \text{if } r_{u,i} \text{ is observed} \\ 0 & \text{otherwise} \end{cases}, \quad (2)$$

where $\Omega_{u,i}$ denotes the u-th row and i-th column entry of Ω. Solving (1) is an ill-posed problem because of overfitting. To address this problem, Tikhonov regularization is widely employed to incorporate into (1) as follows [6]:

$$\underset{\mathrm{X},\,\mathrm{Y}}{\arg\min}\,\varepsilon(\mathrm{X},\,\mathrm{Y}) = \frac{1}{2}\left\|\Omega \odot \left(\mathrm{R} - \mathrm{XY}^{\mathrm{T}}\right)\right\|_{\mathrm{F}}^{2} + \frac{\lambda}{2}\left(\|\mathrm{X}\|_{\mathrm{F}}^{2} + \|\mathrm{Y}\|_{\mathrm{F}}^{2}\right), \quad (3)$$

where λ is the regularization hyperparameter. In general, (3) can be minimized by an optimization algorithm like stochastic gradient descent.

4 Proposed GLFA Model

The proposed GLFA model consists of two main parts. The first part is to construct a weighted interaction graph from a given HiDS matrix to identify its hidden high-confidence HOI. By following the principle of Deep Forest [11] to generate the additional inputs, the second part designs a recurrent LFA structure to recurrently incorporate the identified high-confidence HOI information into the embedding learning process of a graph. Next, we present the two parts, the framework and designed algorithm, and the convergence analysis of the proposed GLFA model, respectively.

4.1 Part 1: Constructing a Graph from an HiDS Matrix to Identify HOI among Nodes.

First, the related definitions regarding part 1 of the proposed GLFA model are given as follows.

***Definition* 3** (*Weighted Interaction Graph of* R). Given an R, let G=(U, I, E, R) be R's weighted interaction graph, where U and I are seen as two types of node sets, E is the edge set, and R is seen as the weight matrix of edges. Each edge $e_{u,i}$ of E denotes that a node u ($u\in U$) connects to a node i ($i\in I$). Each entry $r_{u,i}$ of R denotes the weight of an edge $e_{u,i}$. E can be easily obtained from R. Note that there is no edge between u and i if $r_{u,i}$ is unknown.

***Definition* 4** (*HOI of G*). Given a G, one indirect interaction (u, i) is G's high-order interaction (HOI) if the shortest path from u to i pass through at least p-1 (p≥2) nodes from U. p denotes the p-th HOI.

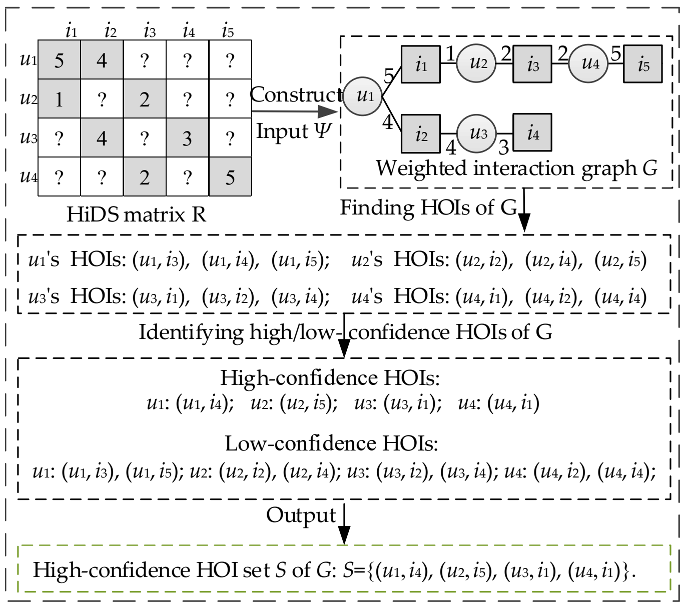
***Definition* 5** (*High/low-confidence HOI*). Given a G, let (u, i) be a p-th HOI. Supposing there are several different paths from u to i. Among each of the several different paths, if there is any node from I is connected with two different weight edges to nodes from U, (u, i) is low-confidence; otherwise, it is high-confidence.

***Definition* 6** (*High-confidence HOI set*). Given a G, the high-confidence HOI set S consists of all the high-confidence HOIs among G.

To illustrate the above definitions, an example is provided in Fig. 2. Given an R with four users and five items (top left corner), its G is constructed as the top right corner of Fig. 2. Based on the G, high/low-confidence HOI can be identified. For example, u_1's HOI items are i_3 (2-nd), i_4 (2-nd), and i_5 (3-nd), where (u_1, i_4) is high-confidence because the weight between u_3 (connected with i_4) and i_2 is 4 that is the same as that between u_1 and i_2, (u_1, i_3) and (u_1, i_5) are low-confidence because from i_3 and i_5 to u_1, i_1 is connected with two different weight edges (weight 5 and weight 1). Then, u_1's high-confidence HOI is (u_1, i_4) and low-confidence HOIs are (u_1, i_3) and (u_1, i_5). Finally, by finding all the high-confidence HOIs for each user, the high-confidence HOI set S of G can be obtained as follows: S={(u_1, i_4), (u_2, i_5), (u_3, i_1), (u_4, i_1)}.

4.2 Part 2: Recurrently Training GLFA with stochastic gradient descent

To explain how to train GLFA, the n-th recurrent training process is described as a general case. First, (3) is extended into the following single latent-factor-dependent form to efficiently train GLFA:



i1 i2 i3 i4 i5
u1 5 4 ? ? ?
u2 1 ? 2 ? ?
u3 ? 4 ? 3 ?
u4 ? ? 2 ? 5
HiDS matrix R
Construct
Input Ψ
Weighted interaction graph G
Finding HOIs of G
u1's HOIs: (u1, i3), (u1, i4), (u1, i5); u2's HOIs: (u2, i2), (u2, i4), (u2, i5)
u3's HOIs: (u3, i1), (u3, i2), (u3, i4); u4's HOIs: (u4, i1), (u4, i2), (u4, i4)
Identifying high/low-confidence HOIs of G
High-confidence HOIs:
u1: (u1, i4); u2: (u2, i5); u3: (u3, i1); u4: (u4, i1)
Low-confidence HOIs:
u1: (u1, i3), (u1, i5); u2: (u2, i2), (u2, i4); u3: (u3, i2), (u3, i4); u4: (u4, i2), (u4, i4);
Output
High-confidence HOI set S of G: S={(u1, i4), (u2, i5), (u3, i1), (u4, i1)}.

Fig. 2. An example of illustrating the related definitions 3–6 on an HiDS matrix from an e-commerce system with four users and five items.

$$\arg\min_{X^n,\ Y^n} \varepsilon(X^n,\ Y^n)=\frac{1}{2}\sum_{r_{u,i}\in\Psi}\left(r_{u,i}-\sum_{\tau=1}^{f}x^n_{u,\tau}y^n_{i,\tau}\right)^2+\frac{\lambda}{2}\left(\left\|X^n\right\|_F^2+\left\|Y^n\right\|_F^2\right),\quad (4)$$

where X^n and Y^n are the embedding matrices at the n-th recurrent training, $x^n_{u,\tau,}$ is X^n's u-th row and τ-th column entry, and $y^n_{i,\tau,}$ is Y^n's i-th row and τ-th column entry, respectively. To incorporate the identified high-confidence HOI information into an LFA model, (4) is transformed as follows.

$$\arg\min_{X^n,\ Y^n} \varepsilon(X^n,\ Y^n)=$$
$$\underbrace{\frac{1}{2}\sum_{r_{u,i}\in\Psi}\left(r_{u,i}-\sum_{\tau=1}^{f}x^n_{u,\tau}y^n_{i,\tau}\right)^2}_{Training\ on\ input\ HiDS\ matrix}+\underbrace{\frac{1}{2}\alpha\sum_{\bar{r}_{u,i}\in\Lambda}\left(\bar{r}_{u,i}-\sum_{\tau=1}^{f}x^n_{u,\tau}y^n_{i,\tau}\right)^2}_{Training\ on\ high\text{-}confidence\ HOIs}+\underbrace{\frac{\lambda}{2}\left(\left\|X^n\right\|_F^2+\left\|Y^n\right\|_F^2\right)}_{Regularization\ term},\quad (5)$$

where α is the aggregation coefficient, Λ is a set that contains all the predictions for the high-confidence HOIs by the previous n-1 times of recurrent training, and $\bar{r}_{u,i}$ is an element of Λ corresponding to unknown $r_{u,i}$. Note that when n=1, Λ is empty. Then, the instant loss $\varepsilon^n_{u,i}$ of (5) is considered on a single element $\forall\bar{r}_{u,i}\in\Lambda$ or $\forall r_{u,i}\in\Psi$ as follows:

$$\begin{cases}\text{if on } r_{u,i}:\varepsilon^n_{u,i}=\dfrac{1}{2}\left(r_{u,i}-\sum_{\tau=1}^{f}x^n_{u,\tau}y^n_{i,\tau}\right)^2+\dfrac{\lambda}{2}\left(\sum_{\tau=1}^{f}\left(x^n_{u,\tau}\right)^2+\sum_{\tau=1}^{f}\left(y^n_{i,\tau}\right)^2\right)\\ \text{if on } \bar{r}_{u,i}:\varepsilon^n_{u,i}=\dfrac{1}{2}\alpha\left(\bar{r}_{u,i}-\sum_{\tau=1}^{f}x^n_{u,\tau}y^n_{i,\tau}\right)^2+\dfrac{\lambda}{2}\left(\sum_{\tau=1}^{f}\left(x^n_{u,\tau}\right)^2+\sum_{\tau=1}^{f}\left(y^n_{i,\tau}\right)^2\right)\end{cases}.\quad (6)$$

Next, stochastic gradient descent is adopted to optimize (6) *w.r.t.* each single embedding element as follows:

$$\forall\tau\in\{1,2,...,f\}, \text{on } r_{u,i} \text{ or } \overline{r}_{u,i}: \begin{cases} x_{u,\tau}^{n} \leftarrow x_{u,d}^{n} - \eta\dfrac{\partial\varepsilon_{u,i}^{n}}{\partial x_{u,\tau}^{n}} \\ y_{i,\tau}^{n} \leftarrow y_{i,d}^{n} - \eta\dfrac{\partial\varepsilon_{u,i}^{n}}{y_{i,\tau}^{n}} \end{cases}, \tag{7}$$

where η is the learning rate. By extending (7), the training rules are obtained as follows:

$$\forall\tau\in\{1,2,...,f\}:$$

$$\begin{cases} \text{if on } r_{u,i}, \begin{cases} x_{u,\tau}^{n} \leftarrow x_{u,\tau}^{n} + \eta y_{i,\tau}^{n}\left(r_{u,i} - \sum\limits_{\tau=1}^{f} x_{u,\tau}^{n} y_{i,\tau}^{n}\right) - \eta\lambda x_{u,\tau}^{n} \\ y_{i,\tau}^{n} \leftarrow y_{i,\tau}^{n} + \eta x_{u,\tau}^{n}\left(r_{u,i} - \sum\limits_{\tau=1}^{f} x_{u,\tau}^{n} y_{i,\tau}^{n}\right) - \eta\lambda y_{i,\tau}^{n} \end{cases} \\ \text{if on } \overline{r}_{u,i}, \begin{cases} x_{u,\tau}^{n} \leftarrow x_{u,\tau}^{n} + \eta y_{i,\tau}^{n}\alpha\left(\overline{r}_{u,i} - \sum\limits_{\tau=1}^{f} x_{u,\tau}^{n} y_{i,\tau}^{n}\right) - \eta\lambda x_{u,\tau}^{n} \\ y_{i,\tau}^{n} \leftarrow y_{i,\tau}^{n} + \eta x_{u,\tau}^{n}\alpha\left(\overline{r}_{u,i} - \sum\limits_{\tau=1}^{f} x_{u,\tau}^{n} y_{i,\tau}^{n}\right) - \eta\lambda y_{i,\tau}^{n} \end{cases} \end{cases}. \tag{8}$$

After each element in Ψ and Λ is trained with (8), X^n and Y^n are learned. Then, they are employed to predict the elements of S^n, where S^n is a subset of S ($S^n \subseteq S$) whose elements are randomly selected from S at the n-th recurrent training. Let $\hat{S}^n$ be a set whose elements are the predictions for the elements of S^n. $\hat{S}^n$ is obtained by $\hat{R}^n = X^nY^{nT}$, where $\hat{r}_{u,i}^n$ is the entry of $\hat{R}^n$ at u-th row and i-th column. Then, $\hat{S}^n$ is input into a nonlinear activation function as follows:

$$\forall\hat{r}_{u,i}^{n}\in\hat{S}^{n}: \overline{r}_{u,i} = \begin{cases} r_{\min} + 1/\left(1+e^{-\hat{r}_{u,i}^{n}}\right) & if\ \hat{r}_{u,i}^{n} < r_{\min} \\ r_{\max} / \left(1+e^{-\hat{r}_{u,i}^{n}}\right) & if\ \hat{r}_{u,i}^{n} > r_{\max} \\ \hat{r}_{u,i}^{n} & otherswise \end{cases}, \tag{9}$$

where r_{min} and r_{max} denote the minimum and maximum values of R, respectively. Obviously, $\hat{r}_{u,i}^n < r_{min}$ or $\hat{r}_{u,i}^n > r_{max}$ is incorrect, (9) has the function of resetting the extremely unreasonable predictions. Let Λ^n be a set to store the outputs of (9), i.e., Λ^n is comprised of all the outputs of (9) at the n-th recurrent training. Then, Λ^n is put into Λ as the input for the next training, i.e.,

$$\Lambda = \Lambda \cup \Lambda^{n}. \tag{10}$$

Finally, after N times of recurrent training, outputting the learned embedding matrices X^N and Y^N, which can be employed to make R's rank-f approximation $\hat{R}$ by $\hat{R}=X^NY^{NT}$.

4.3 The Framework and Algorithm of GLFA

4.3.1 The Framework

Fig. 3 shows the framework of the proposed GLFA model, in which the red dashed boxes denote part 1 and the blue dashed boxes denote part 2. GLFA works as follows:

1) Input: inputting Ψ of R as the initial input;
2) Part 1: identifying high-confidence HOI set by
   i. constructing weighted interaction graph G based on Ψ (definition 3);
   ii. identifying all the high-confidence HOIs based on G (definitions 4 and 5);
   iii. putting all the identified high-confidence HOIs into set S (definition 6);
3) Part 1 at n-th training: randomly selecting elements from S to generate a subset S^n, $S^n \subseteq S$, $S \leftarrow S - S^n$;
4) Part 2 at n-th training:
   i. training embedding matrices X^n and Y^n based on Ψ and Λ, where $n \in \{1, ..., N\}$, N is pre-setting maximum times of recurrent training, and Λ contains all the predictions for the elements in S by the previous n-1 times of recurrent training. Note that Λ is empty when n=1;
   ii. predicting all the elements of S_n via $\hat{R}^n = X^nY^{nT}$, and putting the predictions into set $\hat{S}^n$;
   iii. inputting $\hat{S}^n$ into a nonlinear activation function, its output is set Λ^n;
   iv. putting Λ^n into Λ as the input for the next training;
5) Recurrent training: repeating steps 3)–4) from n=1 to n=N, i.e., recurrently training steps 3)–4) N times;
6) Output: outputting the learned X^n and Y^n.

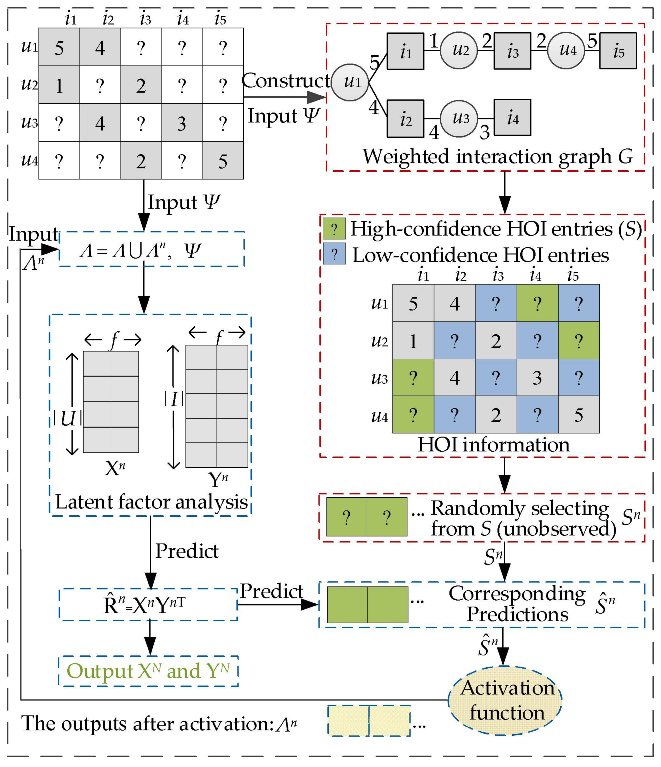


Fig. 3. The proposed GLFA model, in which the red dashed boxes belong to part 1 and the blue dashed boxes belong to part 2. Note that the Λ is continuously expanded during the recurrent training, and both Λ and Ψ are repeatedly input for training a better LFA model during the recurrent training process.

4.3.2 The designed algorithm

Based on the above inferences, the algorithm of GLFA model is designed as shown in *Algorithm* 1. Its time complexity consists of two parts. Let θ be the density of observed entities of R, the time complexity of first part is $O(|U|\times(|U|-1)\times\theta/2)\approx O(|U|^2\times\theta)$ for identifying high-confidence HOI. According to [19], the time complexity of the first recurrent training is $O(T_{max}\times|\Psi|\times f)$, where T_{max} is the maximum iteration count for each recurrent training. Supposing the predicted ratio of S is ν at each recurrent training, where $\nu=|\hat{S}^n|/|S|$. The time complexity of the second part is $O(T_{max}\times|\Psi|\times f\times N(1+N\times\nu\times|S|))$ by recurrently training N times. Although GLFA has a higher time complexity than an LFA model, it is acceptable in practice because 1) the HiDS matrix is commonly highly sparse, and 2) GLFA can be parallelized by Hogwild! [41] to significantly reduce its computational cost. In terms of space complexity, *GLFA* has the following data structures: 1) an array with length $|\Psi|\times(1+\nu)^{N-1}$ to cache the input data for training, 2) a matrix with size $|U|\times f$ to cache X^n, and 3) a matrix with size $|I|\times f$ to cache Y^n. Hence, the space complexity of *Algorithm* 1 *GLFA* is $O(\max\{|\Psi|\times(1+\nu)^{N-1}, |U|\times f, |I|\times f\})$.

ALGORITHM 1 GLFA

```
Steps  Input: $\Psi$; Output: $X^N$ and $Y^N$.
1      initializing $p, f, \lambda, \eta, T_{max}$
2      constructing $G$ based on $\Psi$ (definition 3)
3      identifying all the high-confidence HOIs based on $G$ (definitions 4 and 5)
4      putting all the identified high-confidence HOIs into set $S$ (definitions 6)
5      for $n$=1 to $N$
6        initializing $X^n$, and $Y^n$
7        while $t\le T_{max}$ && not converge
8          for $\forall r_{u,i}\in\Psi$ or $\forall\bar{r}_{u,i}\in\Lambda$
9            for $\tau$=1 to $f$
10             computing $x^n_{u,\tau}$ according to (8)
11             computing $y^n_{i,\tau}$ according to (8)
12           end for
13         end for
14         $t$=$t$+1
15       end while
16       randomly selecting elements from $S$ to form $S^n$, $S^n\subseteq S$, $S\leftarrow S-S_n$.
17       predicting the elements in $S^n$ via $\hat{R}^n=X^nY^{nT}$
18       inputting the prediction set $\hat{S}^n$ into (9) to obtain $\Lambda^n$
19       Putting $\Lambda^n$ into $\Lambda$ as input for next time of recurrent training
20     end for
```

4.4 Convergence Analysis

***Theorem* 1.** Given an HiDS matrix R, a GLFA model is convergent in learning two embedding matrices X and Y to make R′s rank-f approximation $\hat{R}$ by minimizing the distances between R and $\hat{R}$ on observed entries set Ψ of R with stochastic gradient descent.

Proof. Since the non-convex objective function (5) is the sum of the instant loss (6), i.e., $\varepsilon(X^n, Y^n)=\sum_{r_{u,i}\in\Lambda\cup\bar{r}_{u,i}\in\Psi}\varepsilon^n_{u,i}$, some relaxations should be made to make it converge [19] as follows:

a) Instant loss $\varepsilon^n_{u,i}$ is considered to instead of the sum loss $\varepsilon(X^n, Y^n)$ on a single-shot stochastic gradient descent for each update, i.e., the loss is computed on $\forall\bar{r}_{u,i}$ or $\forall r_{u,i}$;

b) One-half of the non-convex term is fixed to make the instant loss $\varepsilon^n_{u,i}$ convex, i.e., $y^n_{i,\cdot}$ is treated as a constant to show the convergence with the update of $x^n_{u,\cdot}$, where $x^n_{u,\cdot}$ and $y^n_{i,\cdot}$ denote the u-th row-vector of X^n and the i-th row-vector of Y^n, respectively. Notably, the convergence with the update of both $y^n_{i,\cdot}$ and $x^n_{u,\cdot}$ are the same.

Next, to prove *Theorem* 1, *Lemma* 1 and *Lemma* 2 should be proofed**.** The first is to define L-smooth and strong convex function [37], respectively.

***Definition* 7** (L-smooth function $f(\xi)$). $f(\xi)$ is L-smooth, if

$$\forall\xi_1,\xi_2\in\mathbb{R}^f\ s.t.\ \left\|\nabla f(\xi_1)-\nabla f(\xi_2)\right\|_2\le L\left\|\xi_1-\xi_2\right\|_2. \quad (11)$$

***Definition* 8** (Strong convex function $f(\xi)$). $f(\xi)$ is strong convex if there exists a constant δ>0 satisfying

$$\forall\xi_1,\xi_2\in\mathbb{R}^f\ s.t.\ f(\xi_1)\ge f(\xi_2)+\nabla f(\xi_2)(\xi_1-\xi_2)^{\mathrm{T}}+\frac{1}{2}\delta\left\|\xi_1-\xi_2\right\|_2^2. \quad (12)$$

***Lemma* 1**. The instant loss $\varepsilon^n_{u,i}$ is L-smooth when L is the maximum singular value for matrix $(y^{n\mathrm{T}}_{i,\cdot}y^n_{i,\cdot}+\lambda E_f)$ and E_f is a $f\times f$ identity matrix.

Proof. Assuming $x_{\sigma,\cdot}$ and $x_{\varphi,\cdot}$ are two independent and arbitrary row-vectors of embedding matrix X^n, then:

$$\begin{cases}\text{if on } r_{u,i}\begin{cases}\nabla\varepsilon^n_{u,i}(x_{\sigma,\cdot})-\nabla\varepsilon^n_{u,i}(x_{\varphi,\cdot})\\ =-(r_{u,i}-x_{\sigma,\cdot}y^n_{i,\cdot})y^n_{i,\cdot}+\lambda x_{\sigma,\cdot}+(r_{u,i}-x_{\varphi,\cdot}y^n_{i,\cdot})y^n_{i,\cdot}-\lambda x_{\varphi,\cdot}\\ =(x_{\sigma,\cdot}-x_{\varphi,\cdot})(y^{n\,\mathrm{T}}_{i,\cdot}y^n_{i,\cdot}+\lambda E_f)\end{cases}\\ \text{if on } \bar{r}_{u,i}\begin{cases}\nabla\varepsilon^n_{u,i}(x_{\sigma,\cdot})-\nabla\varepsilon^n_{u,i}(x_{\varphi,\cdot})\\ =-\alpha(\bar{r}_{u,i}-x_{\sigma,\cdot}y^n_{i,\cdot})y^n_{i,\cdot}+\lambda x_{\sigma,\cdot}+\alpha(\bar{r}_{u,i}-x_{\varphi,\cdot}y^n_{i,\cdot})y^n_{i,\cdot}-\lambda x_{\varphi,\cdot}\\ =(x_{\sigma,\cdot}-x_{\varphi,\cdot})(\alpha y^{n\,\mathrm{T}}_{i,\cdot}y^n_{i,\cdot}+\lambda E_f)\end{cases}\end{cases}. \quad (13)$$

To simplify (13) for convergence analysis, α is set as α=1, then:

$$\left\|\nabla\varepsilon^n_{u,i}(x_{\sigma,\cdot})-\nabla\varepsilon^n_{u,i}(x_{\varphi,\cdot})\right\|_2=\left\|(x_{\sigma,\cdot}-x_{\varphi,\cdot})(y^{n\,\mathrm{T}}_{i,\cdot}y^n_{i,\cdot}+\lambda E_f)\right\|_2. \quad (14)$$

According to the L_2-norm properties of a matrix [38], the following inequality is obtained:

$$\left\|\nabla\varepsilon^n_{u,i}(x_{\sigma,\cdot})-\nabla\varepsilon^n_{u,i}(x_{\varphi,\cdot})\right\|_2\le\left\|(y^{n\,\mathrm{T}}_{i,\cdot}y^n_{i,\cdot}+\lambda E_f)\right\|_2\left\|(x_{\sigma,\cdot}-x_{\varphi,\cdot})\right\|_2, \quad (15)$$

where $\|(y^{n\mathrm{T}}_{i,\cdot}y^n_{i,\cdot}+\lambda E_f)\|_2$ indicates the largest singular value of $(y^{n\mathrm{T}}_{i,\cdot}y^n_{i,\cdot}+\lambda E_f)$. Then, $L=\|(y^{n\mathrm{T}}_{i,\cdot}y^n_{i,\cdot}+\lambda E_f)\|_2$ is achieved. Thus, *Lemma* 1 holds. □

***Lemma* 2**. The instant loss $\varepsilon^n_{u,i}$ is of strong convexity when δ is the minimum singular value for matrix $(y^{n\mathrm{T}}_{i,\cdot}y^n_{i,\cdot}+\lambda E_f)$.

Proof. Given arbitrary vectors $x_{\sigma,\cdot}$ and $x_{\varphi,\cdot}$, we expand the state of $\varepsilon^n_{u,i}$ at $x_{\varphi,\cdot}$. According to the principle of the Taylor series, the following formula is obtained:

$$\begin{aligned}&\varepsilon^n_{u,i}(x_{\sigma,\cdot})\approx\varepsilon^n_{u,i}(x_{\varphi,\cdot})+\nabla\varepsilon^n_{u,i}(x_{\varphi,\cdot})(x_{\sigma,\cdot}-x_{\varphi,\cdot})^{\mathrm{T}}\\ &+\frac{1}{2}(x_{\sigma,\cdot}-x_{\varphi,\cdot})\nabla^2\varepsilon^n_{u,i}(x_{\varphi,\cdot})(x_{\sigma,\cdot}-x_{\varphi,\cdot})^{\mathrm{T}}.\\ &\Rightarrow\varepsilon^n_{u,i}(x_{\sigma,\cdot})-\varepsilon^n_{u,i}(x_{\varphi,\cdot})=\nabla\varepsilon^n_{u,i}(x_{\varphi,\cdot})(x_{\sigma,\cdot}-x_{\varphi,\cdot})^{\mathrm{T}}\\ &+\frac{1}{2}(x_{\sigma,\cdot}-x_{\varphi,\cdot})\nabla^2\varepsilon^n_{u,i}(x_{\varphi,\cdot})(x_{\sigma,\cdot}-x_{\varphi,\cdot})^{\mathrm{T}}.\end{aligned} \quad (16)$$

Based on *Definition* 8, if $\varepsilon_{u,i}^n$ is strongly convex, then:

$$\varepsilon_{u,i}^n\left(x_{\sigma,\cdot}\right)-\varepsilon_{u,i}^n\left(x_{\varphi,\cdot}\right)\geq\nabla\varepsilon_{u,i}^n\left(x_{\varphi,\cdot}\right)\left(x_{\sigma,\cdot}-x_{\varphi,\cdot}\right)^{\mathrm{T}}+\frac{1}{2}\delta\left\|x_{\sigma,\cdot}-x_{\varphi,\cdot}\right\|_2^2. \quad (17)$$

Hence, *Lemma* 2 is equivalent to selecting δ to make the following inequality hold:

$$\left(x_{\sigma,\cdot}-x_{\varphi,\cdot}\right)\nabla^2\varepsilon_{u,i}^n\left(x_{\varphi,\cdot}\right)\left(x_{\sigma,\cdot}-x_{\varphi,\cdot}\right)^{\mathrm{T}}\geq\delta\left\|x_{\sigma,\cdot}-x_{\varphi,\cdot}\right\|_2^2. \quad (18)$$

From the form of $\varepsilon_{u,i}^n$, it can achieve that:

$$\nabla^2\varepsilon_{u,i}^n\left(x_{\varphi,\cdot}\right)=y_{i,\cdot}^{n\,\mathrm{T}}y_{i,\cdot}^n+\lambda\mathrm{E}_f. \quad (19)$$

By combining (19) into (18), it only needs to prove that:

$$\begin{aligned}&\left(x_{\sigma,\cdot}-x_{\varphi,\cdot}\right)\left(y_{i,\cdot}^{n\,\mathrm{T}}y_{i,\cdot}^n+\lambda\mathrm{E}_f\right)\left(x_{\sigma,\cdot}-x_{\varphi,\cdot}\right)^{\mathrm{T}}\geq\delta\left\|\left(x_{\sigma,\cdot}-x_{\varphi,\cdot}\right)\right\|_2^2.\\&\Rightarrow\left(x_{\sigma,\cdot}-x_{\varphi,\cdot}\right)\left(y_{i,\cdot}^{n\,\mathrm{T}}y_{i,\cdot}^n+\lambda\mathrm{E}_f-\delta\mathrm{E}_f\right)\left(x_{\sigma,\cdot}-x_{\varphi,\cdot}\right)^{\mathrm{T}}\geq 0.\end{aligned} \quad (20)$$

Based on the properties of the matrix, (20) is equivalent to prove that ($y_{i,\cdot}^{n\mathrm{T}}y_{i,\cdot}^n$+ λE$_f$-δE$_f$) is a positive semi-definite matrix. As analyzed in [38], ($y_{i,\cdot}^{n\mathrm{T}}$ $y_{i,\cdot}^n$+ λE$_f$ -δE$_f$) is a positive semi-definite matrix when δ is the minimum singular value of ($y_{i,\cdot}^{n\mathrm{T}}y_{i,\cdot}^n$+ λE$_f$), and it satisfies positive semi-definiteness. Therefore, *Lemma* 2 holds. □

At the *t*-th iteration, GLFA has the following updating rule for $x_{u,\cdot}^n$ on a single entry $\forall\bar{r}_{u,i}\in\Lambda$ or $\forall r_{u,i}\in\Psi$:

$$x_{u,\cdot}^n(\omega)\leftarrow x_{u,\cdot}^n(\omega-1)-\eta\cdot\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right). \quad (21)$$

where $x_{u,\cdot}^n$(ω) and $x_{u,\cdot}^n$(ω-1) denote the state of $x_{u,\cdot}^n$ updated by the ω-th and (ω-1)-th entries in the *t*-th iteration, respectively. Let $x_{u,\cdot}^{n,*}$ be the optimal state of $x_{u,\cdot}^n$, it can deduce that

$$\begin{aligned}&\left\|x_{u,\cdot}^n(\omega)-x_{u,\cdot}^{n,*}\right\|_2^2=\left\|x_{u,\cdot}^n(\omega-1)-\eta\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right)-x_{u,\cdot}^{n,*}\right\|_2^2\\&=\left\|x_{u,\cdot}^n(\omega-1)-x_{u,\cdot}^{n,*}\right\|_2^2-2\eta\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right)\left(x_{u,\cdot}^n(\omega-1)-x_{u,\cdot}^{n,*}\right)^{\mathrm{T}}\\&+\eta^2\left\|\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right)\right\|_2^2.\end{aligned} \quad (22)$$

According to *Lemma* 2, it can obtain that

$$\begin{aligned}&\varepsilon_{u,i}^n\left(x_{u,\cdot}^{n,*}\right)-\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right)\geq\\&\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right)\left(x_{u,\cdot}^{n,*}-x_{u,\cdot}^n(\omega-1)\right)^{\mathrm{T}}+\frac{1}{2}\delta\left\|x_{u,\cdot}^{n,*}-x_{u,\cdot}^n(\omega-1)\right\|_2^2.\end{aligned} \quad (23)$$

Since $x_{u,\cdot}^{n,*}$ is the optimal state of $x_{u,\cdot}^n$, it obtains that

$$\begin{cases}\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^{n,*}\right)=0,\\\varepsilon_{u,i}^n\left(x_{u,\cdot}^{n,*}\right)<\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right).\end{cases} \quad (24)$$

By incorporating (24) into (23), it has that

$$\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right)\left(x_{u,\cdot}^n(\omega-1)-x_{u,\cdot}^{n,*}\right)^{\mathrm{T}}\geq\frac{1}{2}\delta\left\|x_{u,\cdot}^n(\omega-1)-x_{u,\cdot}^{n,*}\right\|_2^2. \quad (25)$$

Hence, based on (25), (22) is transformed to

$$\left\|x_{u,\cdot}^n(\omega)-x_{u,\cdot}^{n,*}\right\|_2^2\leq\left(1-\eta\delta\right)\left\|x_{u,\cdot}^n(\omega-1)-x_{u,\cdot}^{n,*}\right\|_2^2+\eta^2\left\|\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right)\right\|_2^2. \quad (26)$$

Then, the expectation of (26) is:

$$\begin{aligned}&\mathbb{E}\left[\left\|x_{u,\cdot}^n(\omega)-x_{u,\cdot}^{n,*}\right\|_2^2\right]\\&\leq\left(1-\eta\delta\right)\mathbb{E}\left[\left\|x_{u,\cdot}^n(\omega-1)-x_{u,\cdot}^{n,*}\right\|_2^2\right]+\eta^2\mathbb{E}\left[\left\|\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right)\right\|_2^2\right].\end{aligned} \quad (27)$$

According to [39], we assume that there is a positive number z satisfying that

$$\mathbb{E}\left[\left\|\nabla\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega-1)\right)\right\|_2^2\right]\leq z^2. \quad (28)$$

Then, (27) is transformed to

$$\mathbb{E}\left[\left\|x_{u,\cdot}^n(\omega)-x_{u,\cdot}^{n,*}\right\|_2^2\right]\leq\left(1-\eta\delta\right)\mathrm{E}\left[\left\|x_{u,\cdot}^n(\omega-1)-x_{u,\cdot}^{n,*}\right\|_2^2\right]+\eta^2z^2. \quad (29)$$

By setting the learning rate *η=β/(δt)* with *β*>1, (29) is transformed to:

$$\mathbb{E}\left[\left\|x_{u,\cdot}^n(\omega)-x_{u,\cdot}^{n,*}\right\|_2^2\right]\leq\left(1-\frac{\beta}{t}\right)\mathbb{E}\left[\left\|x_{u,\cdot}^n(\omega-1)-x_{u,\cdot}^{n,*}\right\|_2^2\right]+\frac{1}{t^2}\left(\frac{\beta z}{\delta}\right)^2. \quad (30)$$

By expanding (30) with induction, there is a bound that

$$\mathbb{E}\left[\left\|x_{u,\cdot}^n(\omega)-x_{u,\cdot}^{n,*}\right\|_2^2\right]\leq\frac{1}{t}\max\left\{\left\|x_{u,\cdot}^n(1)-x_{u,\cdot}^{n,*}\right\|_2^2,\frac{\beta^2z^2}{\delta\beta-1}\right\}, \quad (31)$$

where $x_{u,\cdot}^n$(1) represents the initial state of $x_{u,\cdot}^n$ at the *t*-th iteration. Since $\varepsilon_{u,i}^n$ is *L*-smooth by *Lemma* 1, it can achieve that

$$\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega)\right)-\varepsilon_{u,i}^n\left(x_{u,\cdot}^{n,*}\right)\leq\frac{L}{2}\left\|x_{u,\cdot}^n(\omega)-x_{u,\cdot}^{n,*}\right\|_2^2. \quad (32)$$

By computing the expectation of (32), it has that

$$\mathbb{E}\left[\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega)\right)-\varepsilon_{u,i}^n\left(x_{u,\cdot}^{n,*}\right)\right]\leq\frac{L}{2}\mathbb{E}\left[\left\|x_{u,\cdot}^n(\omega)-x_{u,\cdot}^{n,*}\right\|_2^2\right]. \quad (33)$$

By incorporating (31) into (33), it can deduce that

$$\mathbb{E}\left[\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega)\right)-\varepsilon_{u,i}^n\left(x_{u,\cdot}^{n,*}\right)\right]\leq\frac{L}{2t}\Theta\left(\beta\right), \quad (34)$$

where it has

$$\Theta\left(\beta\right)=\max\left\{\left\|x_{u,\cdot}^n(1)-x_{u,\cdot}^{n,*}\right\|_2^2,\frac{\beta^2z^2}{\delta\beta-1}\right\}. \quad (35)$$

Then, to expand (34) on all the known entries of Λ and Ψ:

$$\mathbb{E}\left[\sum_{(u,i)\in\Lambda\cup\Psi}\left(\varepsilon_{u,i}^n\left(x_{u,\cdot}^n(\omega)\right)-\varepsilon_{u,i}^n\left(x_{u,\cdot}^{n,*}\right)\right)\right]\leq\left(\left|\Lambda\right|+\left|\Psi\right|\right)\frac{L}{2t}\Theta\left(\beta\right), \quad (36)$$

where it has $\left(\left|\Lambda\right|+\left|\Psi\right|\right)\frac{L}{2t}\Theta\left(\beta\right)\rightarrow 0$, when $t\rightarrow\infty$.

Notably, although (5) is non-convex, $x_{u,\cdot}^n$ and $y_{i,\cdot}^n$ can be updated alternatively by stochastic gradient descent. Hence, when $x_{u,\cdot}^n$ is treated as a constant, the situation is the same. In addition, as analyzed in [40], stochastic gradient descent requires learning rate $\eta\leq 1/\delta t$ at the *t*-th iteration, which is also guaranteed by *Lemma* 2. Besides, the regularization does not impact the convergence [40]. Therefore, *Theorem* 1 holds.□

5 Experiments

The experiments aim at answering the following research questions (RQs):

RQ. 1. Does the proposed GLFA model outperform state-of-the-art models in representing an HiDS matrix?

RQ. 2. Is the HOI information of an HiDS matrix helpful for improving GLFA's representation learning ability?

RQ. 3. How does the recurrent LFA structure influence GLFA's performance?

5.1 General Settings

Datasets. Three benchmarks of real-world HiDS datasets are tested in the experiments. Their properties are summarized in Table 2. Ml1m is collected by the MovieLens[1] system which is an online movie platform. Yahoo is collected by Yahoo!Music[2] and records a music community's preferences for various songs. Goodbook[3] is collected by a real-world online book site with six million ratings for ten thousand most popular books. The 20% known data of each dataset are randomly selected as the training set, and the remaining 80% are treated as the testing set.

TABLE 2. PROPERTIES OF ALL THE HIDS DATASETS.

Name	$\|U\|$	$\|I\|$	$\|\Psi\|$	Density*
ML1M[1]	6040	3706	1,000,209	4.47%
Yahoo[2]	15400	1000	365,704	2.37%
Goodbooks[3]	53424	10000	5,976,480	1.12%

*Density denotes the percentage of known data in the HiDS matrix.

Evaluation Protocol. In many big-data-related applications, it is highly significant to recover the unknown relationships among an HiDS matrix. Hence, the task of missing data estimation is widely adopted as the evaluation protocol of representing an HiDS matrix [8, 13].To this end, root mean squared error (RMSE) and mean absolute error (MAE) are usually adopted [13] to evaluate the prediction accuracy as:

$$RMSE=\sqrt{\left(\sum_{(w,j)\in\Upsilon}\left(r_{w,j}-\hat{r}_{w,j}\right)^2\right)\Big/\left|\Upsilon\right|},$$

$$MAE=\left(\sum_{(w,j)\in\Upsilon}\left|r_{w,j}-\hat{r}_{w,j}\right|_{abs}\right)\Big/\left|\Upsilon\right|,$$

where $|\cdot|_{abs}$ indicate the absolute value and Υ indicates the testing set.

Baselines. GLFA is compared with the following state-of-the-art baselines, including four LFA-based models (BLF, NLF, FNLF, and GFNLF), two DNNs-based models (AutoRec and MetaMF), and two GNNs-based models (MCGC and IGMC).

-BLF [6]. It is the classic LFA model and has been widely adopted to represent HiDS data. It is still highly competitive with DNNs-based models, which is demonstrated by [36].

-NLF [5]. It introduces the nonnegative constraints into an LFA mode by controlling its learning rate. It is highly effective in representing nonnegative HiDS data.

-**FNLF** [13]. It incorporates the momentum into an NLF model to enhance the representation learning ability and computational efficiency.

-GFNLF [20]. It is a generalized and fast-converging NLF model. It adopts the α-β-divergence to construct its objective function to enhance the representation learning ability.

-AutoRec [24]. It employs the autoencoder to construct a collaborative filtering framework. It is the representative DNNs-based model to represent HiDS data from recommender systems.

-DCCR [26]. It is a deep collaborative conjunctive architecture consisting of two different kinds of neural networks. Concretely, it employs an autoencoder and a multilayered perceptron to represent the HiDS data.

-**MCGC** [32]. It is a multi-component graph convolutional collaborative filtering approach. It can distinguish the latent graph structure underneath the HiDS data.

-IGMC [31]. It is an inductive graph-based matrix completion model. It trains a GNN based purely on 1-hop subgraphs generated from the HiDS matrix and maps these subgraphs to make predictions.

Notably, since GLFA aims to estimate the missing data of an HiDS matrix, it mainly compares with matrix completion models. We leave out the comparison with two kinds of models: 1) item ranking models, such as NeuMF [22], NGCF [33], and LightGCN [34], because they have different evaluation metrics, and 2) hybrid models, such as LMF-PP [15], OFRR [18], and DRRNN [30], because they require the additional information like review comments or geographical location.

Implementation Details. To draw a fair comparison, f is set as f=10 for GLFA and the four LFA-based models. For GLFA, default α=1 and N=20. The two DNNs-based and two GNNs-based models set their hyperparameters according to their original papers. Besides, the learning rate and regularization coefficient are tuned for all the models to ensure they can achieve their own best prediction accuracy. Each dataset is pre-treated to construct the HOI graph with default p=2. Each model is tested on the same split training-testing sets of datasets to guarantee a fair comparison. The training termination condition of each model is to reach a preset epoch threshold (e.g., 1000) or a preset error difference between two consecutive iterations (e.g., 10^{-6}). Each model is tested five times and the average results are reported. All the experiments are run on the CPU of a computer server with 2.4 GHz E5-2680, 16 cores, and 64 GB RAM.

5.2 Performance Comparison (RQ1)

5.2.1 Comparison of prediction accuracy

Table 3 presents the comparison results. Besides, the statistical analysis of the win/loss counts of GLFA versus comparison models is conducted to better understand these comparison results. The win/loss counts are summarized in the last row of Table 4. The following three observations can be noted: 1) GLFA wins the four LFA-based models in most cases (loses only one case), 2) when compared with the two

[1] https://grouplens.org/datasets/movielens/
[2] https://webscope.sandbox.yahoo.com/catalog.php?datatype=r
[3] http://fastml.com/goodbooks-10k-a-new-dataset-for-book-recommendations/

TABLE 3. THE COMPARISON RESULTS OF RMSE/MAE, INCLUDING WIN/LOSS COUNTS, WHERE • INDICATES GLFA HAS A LOWER RMSE/MAE THAN THE COMPARISON MODELS.

Dataset	Metric*	BLF	NLF	FNLF	GFNLF	AutoRec	DCCR	MCGC	IGMC	GLFA
Ml1m	RMSE	0.9175●	0.9290●	0.9282●	0.9266●	0.9169●	0.9151●	0.9351●	0.9145●	0.9126
	MAE	0.7243●	0.7336●	0.7280●	0.7264●	0.7271●	0.7253●	0.7454●	0.7283●	0.7205
Yahoo	RMSE	1.4371●	1.4302●	1.4313●	1.4303●	1.4428●	1.4382●	1.3362	1.3062	1.4188
	MAE	1.0844●	1.0868●	1.0831●	1.0861●	1.0925●	1.0899●	1.0609	1.0978●	1.0794
Goodbooks	RMSE	0.8672	0.8711●	0.8701●	0.8731●	0.8790●	0.8758●	0.8861●	0.8631	0.8674
	MAE	0.6812●	0.6778	0.6784	0.6792	0.6802●	0.6824●	0.7024●	0.6866●	0.6800
Statistic	Win/Loss	5/1	5/1	5/1	5/1	6/0	6/0	4/2	4/2	—
	F-rank*	4.33	5.33	4.67	4.83	6.50	5.83	6.50	4.67	2.33

* A **lower** F-rank value denotes a higher prediction accuracy.

TABLE 4. RESULTS OF WILCOXON SIGNED-RANKS TEST ON RMSE/MAE OF TABLE 3.

Comparison	*R+*	*R-*	*p*-value*
GLFA vs. BLF	20	1	**0.0313**
GLFA vs. NLF	20	1	**0.0313**
GLFA vs. FNLF	20	1	**0.0313**
GLFA vs. GFNLF	20	1	**0.0313**
GLFA vs. AutoRec	21	0	**0.0156**
GLFA vs. DCCR	21	0	**0.0156**
GLFA vs. MCGC	14	7	0.2813
GLFA vs. IGMC	13	8	0.3438

* The accepted hypotheses with a significance level of 0.05 are highlighted.

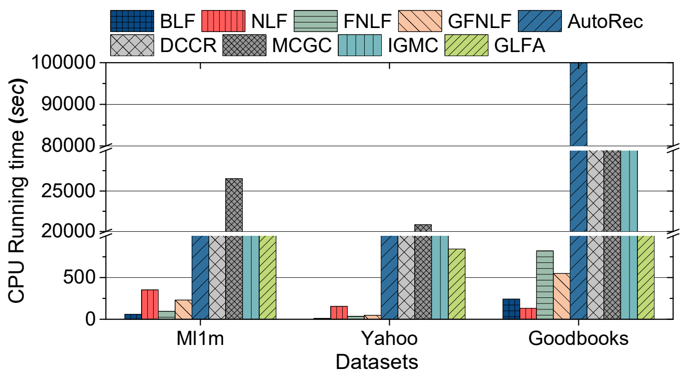


Fig. 4. The comparison of CPU running time.

DNNs-based models, GLFA evidently wins them, 3) GLFA outperforms the two GNNs-based models in most cases. Therefore, these observations show that GLFA has a lower RMSE/MAE than the six comparison models.

Moreover, to check whether GLFA significantly outperforms every single baseline, the Wilcoxon signed-ranks test [21] is conducted on the comparison results of Table 3. Wilcoxon signed-ranks test has three indicators—*R*+, *R*-, and *p*-value. The larger *R*+ value denotes a higher prediction accuracy. The *p*-value denotes the significance level. The statistical results are recorded in Table 4. Supported by statistical evidence, six hypotheses are accepted, i.e., GLFA has significantly higher prediction accuracy than BLF, NLF, FNLF, GFNLF, AutoRec, and MetaMF. We believe that the high prediction accuracy of GLFA originates from its recurrent LFA structure that incorporates the HOI information of an HiDS matrix. Notably, since MCGC and IGMC also consider the graph structure of an HiDS matrix, the two hypotheses of GLFA vs. MCGC and GLFA vs. IGMC are not accepted. However, GLFA still achieves much higher *R*+ values than MCGC and IGMC, supporting that GLFA slightly outperforms them. The main reason is that GLFA employs the highly competitive LFA model [36] as its basic component rather than the neural networks employed by MCGC and IGMC.

5.2.1 Comparison of computational efficiency

To evaluate the computational efficiency of each comparison model, their CPU running time is measured on all the datasets. Fig. 4 records results, where we have the following observations. First, GLFA costs more CPU running time than the four LFA-based models. One reason is that GLFA trains some extra data due to its recurrent LFA structure. However, compared with the two DNNs-based and two GNNs-based models, GLFA costs much less CPU running time than them. The main reason is that GLFA only takes the observed entries of the HiDS matrix as input while they take the complete data.

5.3 The Usefulness of HOI Information for GLFA (RQ2)

This set of experiments tests GLFA's prediction accuracy as aggregation coefficient α increases. With a larger α, more HOI information is incorporated into GLFA. Fig. 5 records the results on Yahoo, where we observe that the RMSE/MAE decreases as α increases at the beginning. After reaching a certain threshold, as α continues to increase, the RMSE/MAE increases. For example, the MAE is reduced from 1.0826 to 1.0727 when α increases from 0 to 50. After that, it keeps growing to 1.0842 as α increases to 200. This observation demonstrates that the HOI information helps improve GLFA's representation learning ability to an HiDS matrix. However, it is not necessary to incorporate superfluous HOI information into GLFA.

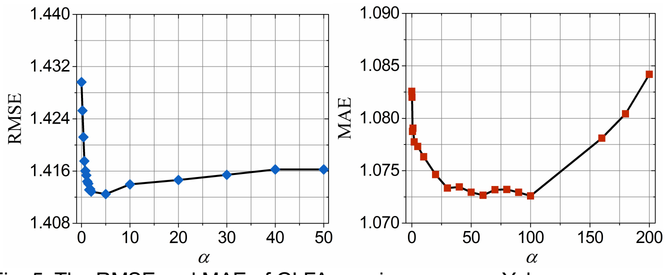


Fig. 5. The RMSE and MAE of GLFA as α increases on Yahoo.

5.4 Influence of Recurrent LFA Structure in GLFA (RQ3)

This set of experiments tests the changes of RMSE/MAE *w.r.t.* the times of recurrent training. Fig. 6 presents the iteration process at different *n*-th recurrent training on Yahoo. From it, we have the following important findings:
1) GLFA's recurrent training structure does not affect an LFA model's (its basic component) convergence. At each recurrent training, the RMSE/MAE keeps decreasing with more iteration counts until reaching convergence.
2) GLFA's recurrent training structure can enhance an LFA model's representation learning ability. With more times of recurrent training, GLFA has a lower RMSE/MAE, i.e., GLFA's prediction accuracy is improved. Notably, GLFA degenerates into a basic LFA model if it recurrently trains only once.
3) GLFA's representation learning ability is not always enhanced with more times of recurrent training. Fig. 6 shows that the MAE slightly increases after the 15th recurrent training. The reason is that the extra recurrent training increases the probability of overfitting that causes performance degradation.

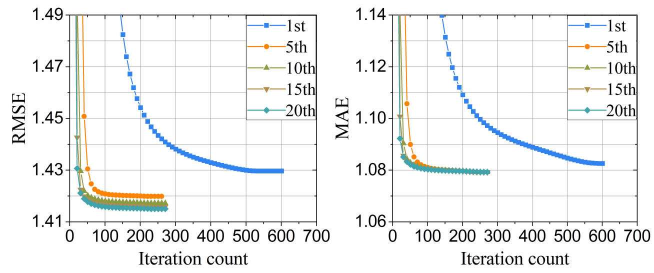


Fig. 6. The iteration process of GLFA at different *n*-th recurrent training on Yahoo.

5.5 Summary of Experiments

Based on the experimental results, GLFA's disadvantages and advantages are summarized as follows.

Disadvantages:
1) To achieve a better performance, GLFA is required to tune its hyperparameter α. This issue is addressed by pre-tuning on warming-up datasets currently. We plan to make α self-adaptive in the future.
2) Compared with an LFA model, GLFA costs extra computations to find the High-confidence HOI set. However, such extra computations can be significantly reduced through parallelization by following Hogwild! [41]

Advantages:
1) GLFA achieves higher accuracy for predicting missing data of an HiDS than its peers do. In detail, GLFA significantly outperforms the four LFA-based and two DNNs-based models, and slightly outperforms the two GNNs-based models.
2) Although GLFA's computational efficiency is lower than the four LFA-based models, it is much higher than that of the two DNNs-based and two GNNs-based models.
3) The appropriate HOI information helps improve GLFA's representation learning ability to an HiDS matrix.
4) GLFA's recurrent training structure does not affect an LFA model's convergence and can enhance its representation learning ability to an HiDS matrix.

6 Conclusion

This paper proposes a graph-incorporated latent factor analysis (GLFA) model for accurately representing an HiDS matrix. By recurrently incorporating high-order interaction (HOI) information among nodes, which is identified from a weighted interaction graph of the HiDS matrix, into GLFA's embedding learning process, its representation learning ability is enhanced. In the experiments, GLFA is compared with eight state-of-the-art models on three real-world datasets. The results demonstrate that it outperforms all the comparison models in predicting the missing data of an HiDS matrix. Moreover, GLFA achieves a much higher computational efficiency than the two deep neural networks (DNNs)-based and two graph neural networks (GNNs)-based models. In the future, we plan to modify GLFA based on an intelligent optimization algorithm to make its hyperparameters self-adaptive.

References

[1] D. M. Camacho, K. M. Collins, R. K. Powers, J. C. Costello, and J. J. Collins, "Next-generation machine learning for biological networks," *Cell*, vol. 173, no. 7, pp. 1581-1592, 2018.
[2] P. Zhang, F. Xiong, H. Leung, and W. Song, "FunkR-pDAE: Personalized Project Recommendation Using Deep Learning," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 886-900, 2021.
[3] X. Wei, Z. Li, Y. Liu, S. Gao, and H. Yue, "SDLSC-TA: Subarea Division Learning Based Task Allocation in Sparse Mobile Crowdsensing," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1344-1358, 2021.
[4] A. Al-Hilo, D. Ebrahimi, S. Sharafeddine, and C. Assi, "Vehicle-Assisted RSU Caching Using Deep Reinforcement Learning," *IEEE Transactions on Emerging Topics in Computing*, pp. 1-1, 2021. DOI: 10.1109/TETC.2021.3068014
[5] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, " A Nonnegative Latent Factor Model for Large-Scale Sparse Matrices in Recommender Systems via Alternating Direction Method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 579-592, 2016.

[6] Y. Koren, R. Bell, and C. Volinsky, “Matrix Factorization Techniques for Recommender Systems,” *Computer,* vol. 42, no. 8, pp. 30-37, 2009.

[7] H. Zhou, G. Yang, Y. Xiang, Y. Bai and W. Wang, "A Lightweight Matrix Factorization for Recommendation With Local Differential Privacy in Big Data," *IEEE Transactions on Big Data*, vol. 9, no. 1, pp. 160-173, 1. 2023.

[8] X. He, J. Tang, X. Du, R. Hong, T. Ren, and T.-S. Chua, “Fast Matrix Factorization With Nonuniform Weights on Missing Data,” *IEEE Transactions on Neural Networks and Learning Systems,* vol. 31, no. 8, pp. 2791-2804, 2020.

[9] W. Cheng, Y. Shen, Y. Zhu, and L. Huang, "DELF: A Dual-Embedding based Deep Latent Factor Model for Recommendation," *In Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI*, 2018, pp. 3329-3335.

[10] R. v. d. Berg, T. N. Kipf, and M. Welling, “Graph Convolutional Matrix Completion,” *arXiv preprint arXiv:1706.02263.*, 2017.

[11] Z.-H. Zhou, and J. Feng, "Deep forest: Towards an alternative to deep neural networks," *In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI* 2017, pp. 3553–3559.

[12] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, and L. Sun, “Dual-regularized matrix factorization with deep neural networks for recommender systems,” *Knowledge-Based Systems,* vol. 145, pp. 46-58, 2018.

[13] X. Luo, Z. Liu, S. Li, M. Shang, and Z. Wang, “A Fast Non-Negative Latent Factor Model Based on Generalized Momentum Method,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 51, no. 1, pp. 610 - 620, 2021.

[14] X. Ren, M. Song, E. Haihong, and J. Song, “Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation,” *Neurocomputing,* vol. 241, pp. 38-55, 2017.

[15] D. Ryu, K. Lee, and J. Baik, “Location-based Web Service QoS Prediction via Preference Propagation to address Cold Start Problem,” *IEEE Transactions on Services Computing*, vol. 14, no. 3, pp. 736-746, 2021.

[16] H. Liu, L. Jing, J. Yu, and M. K. Ng, “Social recommendation with learning personal and social latent factors,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 7, pp. 2956-2970, 2021.

[17] C. Wang, Q. Liu, R. Wu, E. Chen, C. Liu, X. Huang, and Z. Huang, "Confidence-aware matrix factorization for recommender systems," *In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 434-442.

[18] H. Zhang, Y. Sun, M. Zhao, T. W. S. Chow, and Q. M. J. Wu, “Bridging User Interest to Item Content for Recommender Systems: An Optimization Model,” *IEEE Transactions on Cybernetics*, vol. 50, no. 10, pp. 4268-4280, 2020.

[19] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and M. Zhou, “A Deep Latent Factor Model for High-Dimensional and Sparse Matrices in Recommender Systems,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 51, no. 7, pp. 4285-4296, 2021.

[20] Y. Yuan, X. Luo, M. Shang, and D. Wu, “A Generalized and Fast-converging Non-negative Latent Factor Model for Predicting User Preferences in Recommender Systems,” in Proceedings of The Web Conference 2020, Taipei, Taiwan, 2020, pp. 498–507.

[21] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, “A data-characteristic-aware latent factor model for web services QoS prediction,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2525-2538, 2022.

[22] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," *In Proceedings of the 26th International World Wide Web Conference*, 2017, pp. 173-182.

[23] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Computing Surveys (CSUR),* vol. 52, no. 1, pp. 5:1-5:38, 2019.

[24] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec:Autoencoders Meet Collaborative Filtering," *In Proceedings of the 24th International Conference on World Wide Web*, ACM, 2015, pp. 111-112.

[25] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, “Variational Autoencoders for Collaborative Filtering,” in In Proceedings of the 2018 World Wide Web Conference on World Wide Web, ACM WWW, Lyon, France, 2018, pp. 689-698.

[26] Q. Wang, B. Peng, X. Shi, T. Shang, and M. Shang, “DCCR: Deep Collaborative Conjunctive Recommender for Rating Prediction,” *IEEE Access*, vol. 7, pp. 60186-60198, 2019.

[27] X. He, and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 355-364.

[28] Y. Lin, P. Ren, Z. Chen, Z. Ren, D. Yu, J. Ma, M. d. Rijke, and X. Cheng, "Meta Matrix Factorization for Federated Rating Predictions," *In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 981-990.

[29] Y. Tay, L. Anh Tuan, and S. C. Hui, "Latent relational metric learning via memory-based attention for collaborative ranking," *In Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 729-739.

[30] W. D. Xi, L. Huang, C. D. Wang, Y. Y. Zheng, and J. H. Lai, “Deep Rating and Review Neural Network for Item Recommendation,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 11, pp. 6726-6736, 2022.

[31] M. Zhang, and Y. Chen, "Inductive Matrix Completion Based on Graph Neural Networks," *In Proceeding of the 8th International Conference on Learning Representations, ICLR*, 2020, pp. 262-269.

[32] X. Wang, R. Wang, C. Shi, G. Song, and Q. Li, "Multi-component graph convolutional collaborative filtering," *In Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 6267-6274.

[33] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural Graph Collaborative Filtering," *In Proceedings of the 42nd International Conference on Research and Development in Information Retrieval, ACM SIGIR* 2019, pp. 165-174.

[34] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," *In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639-648.

[35] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," *In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 726-735.

[36] S. Rendle, W. Krichene, L. Zhang, and J. R. Anderson, "Neural Collaborative Filtering vs. Matrix Factorization Revisited," *In Proceedings of the 14th ACM Conference on Recommender Systems, RecSys*, 2020, pp. 240-248.

[37] S. Boyd, and L. Vandenberghe, “Convex optimization,” *Cambridge: Cambridge University Press*, 2009.

[38] X. D. Zhang, “Matrix analysis and applications,” *Cambridge University Press*, 2017.

[39] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, “Robust stochastic approximation approach to stochastic programming,” *SIAM Journal on optimization,* vol. 19, no. 4, pp. 1574-1609, 2009.

[40] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," *In Proceedings of*

the International Conference on Machine Learning, ICML, 2012, pp. 1571-1578.

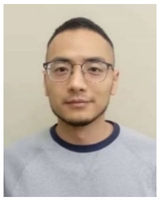
[41] B. Recht, C. Ré, S. J. Wright, and F. Niu, "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent," *In Proceedings of the 25th Annual Conference on Neural Information Processing Systems, NIPS*, 2011, pp. 693-701.

[42] W Zhang, Z Yin, Z Sheng, Y Li, W Ouyang, X Li, Y Tao, Z Yang, and B Cui "Graph attention multi-layer perceptron," *In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4560-4570.

[43] S Wu, Y Zhang, C Gao, K Bian, and B Cui, "Garg: anonymous recommendation of point-of-interest in mobile networks by graph convolution network," *Data Science and Engineering*, vol 5, pp.433-447, 2020.

[44] Y Zhang, F Sun, X Yang, C Xu, W Ou, and Yan Zhang, "Graph-based Regularization on Embedding Layers for Recommendation," *ACM Transactions on Information Systems*, vol 39, no.1, pp. 1-27, 2020.

[45] S Wu, F Sun, W Zhang, X Xie, and B Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys*, vol 55, no.5, pp. 1-37, 2023.



Di Wu (Member, IEEE) received his Ph.D. degree from the Chongqing Institute of Green and Intelligent Technology (CIGIT), Chinese Academy of Sciences (CAS), China in 2019. He is currently a Professor of the College of Computer and Information Science, Southwest University, Chongqing, China. He has over 70 publications, including 15 IEEE TRANSACTIONS papers and several conference papers on AAAI, ICDM, WWW, IJCAI, etc. His research interests include machine learning and data mining. He is serving as an Associate Editor for the Neurocomputing (IF 5.779) and Frontiers in Neurorobotics (IF 3.493). For more information please visit his homepage: https://wudi1989.github.io/Homepage/

Yi He (Member, IEEE) received his Ph.D. degree in computer science from the University of Louisiana at Lafayette and a B.E. from the Harbin Institute of Technology (China) in 2020 and 2013, respectively. Dr. Yi He is an assistant professor of Computer Science at ODU. His research focus lies broadly in data mining and machine learning and specifically in online learning, data stream analytics, and graph theory. His research outcomes have appeared in top venues, e.g., AAAI, IJCAI, WWW, ICDM, SDM, TKDE, TNNLS, to name a few. He has served as multiple roles in the data mining and machine learning community, including the registration chair of ICDM 2022, session chair of ICMR 2022, and (T)PC member of AAAI'20,'21,'22, IJCAI'22, ICDM'22, CIKM'22, ECML-PKDD'22, and referees for prestigious journal such as TNNLS, TMC, and TITS.

Xin Luo (Senior Member, IEEE) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from the Beihang University, Beijing, China, in 2011. He is currently a Professor of Data Science and Computational Intelligence with the College of Computer and Information Science, Southwest University, Chongqing, China. He has authored or coauthored over 200 papers (including over 110 IEEE Transactions papers) in the areas of his interests. His research interests focus on big data analysis and graph learning. Dr. Luo was the recipient of the Outstanding Associate Editor Award from IEEE/CAA JOURNAL OF AUTOMATICA SINICA in 2020. He is currently serving as a Deputy-Editor-in-Chief for IEEE/CAA JOURNAL OF AUTOMATICA SINICA, and an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS. His Google page: https://scholar.google.com/citations?user=hyGlDs4AAAAJ&hl=zh-TW.